UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/808,905	03/25/2004	Anson Horton	13768.1256	7406

47973 7590 10/02/2009
WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111

EXAMINER	
SMITH, CHENECA	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
10/02/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/808,905	**Applicant(s)** HORTON ET AL.	
	Examiner CHENECA P. SMITH	**Art Unit** 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _09 July 2009_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-19,21 and 22_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-19,21 and 22_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _25 March 2004_ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on July 9, 2009 has been entered.

2. Claims 1-19, 21 and 22 have been examined.

3. Applicant's arguments have been fully considered but they are not persuasive.

Response to Arguments

4. Applicant's arguments filed July 9, 2009 have been fully considered but they are not persuasive.

In response to applicant's arguments regarding claims 1, 17 and 22 that " the evaluation of attributes by the expression evaluator results in the display of information that would otherwise not be displayed; Bates and Bates_2 considered both alone and in combination do not teach this element" (see page 10 1st paragraph), applicant should note where Bates teaches configuring the user interface screen with viewable attributes indicative of each instance of a variable in at least the code displayed in the source code panel (see [0010] lines 6-9). Bates also discloses where the debugger determines whether the variable value is associated with a field of a record or whether a field is to be displayed (*i.e. a comment associated with the variable*) as opposed to a normal singular variable (see [0065]). Therefore, Bates does teach attributes that allow additional information about the variables in an application to be displayed to a user.

In response to applicant's argument regarding claims 1, 17 and 22 that "Bates_2 in combination with Bates does not teach the ability to evaluate an expression contained in the code "(see page 11, 1st paragraph), Applicant should note where Bates discloses where the debugger determines whether any attributes are set for the variables (see [0064]) and whether any internal comments exist for the variable, which are embedded within the source code (see [0063]). Bates also discloses that the debugger determines whether the variable value is associated with a field of a record or whether a field is to be displayed (*i.e. a comment associated with the variable*) as opposed to a normal singular variable (see [0065]). Therefore, Bates does teach the evaluation of expressions contained in the code being debugged.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1, 5-7, 13-17-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bates et al (US Patent Application Publication 2003/0221185 A1) in view of Bates et al (US Patent 7,251,808 B2, referred to as Bates_2, both arts already of record).

As to claim 1, Bates teaches a computing system to facilitate attributed debugging system for debugging a computer software application, the computing system comprising:

A processor (see Fig.1, 112 and associated text), and

a computer readable storage medium (see [0023]) storing the following components:

a debugger that facilitates debugging of a computer software application (see FIG.1: 123 and associated text, e.g. [0040]),

a debuggee that includes one or more attributes associated with the computer software application, which attributes are employed by the debugger to facilitate debugging of the software application (see Fig, 1 119 and associated text, e.g. [0037] lines 8-10) and,

an expression evaluator (see Fig.1, 126 and associated text) that evaluates the one or more attributes associated with the computer software application according to an attribute definition (see Fig. 3 and associated text, e.g. [0047] lines 23-31), and presents debug information associated with the computer software application in accordance with the attribute definition (see Fig.6B, 628 and associated text, e.g. [0067] lines 8-10), wherein the evaluation of at least one of the one or more attributes (*i.e. evaluation of Call attribute to be on* *or off*, see Fig.3 and associated text) determines that a value (*i.e. character C*, see Fig.7, 710 and associated text) to be shown for a field or property (*i.e. flyover text box,* see FIG.7, 710 and associated text, e.g. [0067] lines 12-14) of the software application is the result of an evaluation of an expression contained in the code of the software application (*i.e. determination that the Call attribute is to be on*, see Fig. 3 and associated text), which value is different from the value that would be shown for the field or property when the corresponding attribute is removed (*i.e. attribute is turned off*, see Fig. 3 and associated text and *therefore not displayed in the flyover text box*, see Fig.7,710 and associated text).

Bates does not specifically teach wherein the attribute definition declaratively indicates tow-the developer-customizable format in which the debug information is displayed in a developer-customizable data window. In an analogous art, however, Bates_2 is cited to teach where debug information is presented in a developer-customizable format (see column 2, lines 8-12 and lines 28-37 and FIG. 4 and associated text, e.g. column 3, lines 49-55). It would have been obvious to one having ordinary skill in the art at the time of the invention to combines the teachings of Bates and Bates_2 for the purpose of providing an improved debugger having a mechanism when debugging programs that provides custom record displays, where only user selected fields of records or variables are displayed, as disclosed by Bates_2 (see column 1, lines 53-56).

As to claim 5, Bates in view of Bates_2 teaches the limitations of claim 1, but does not specifically teach that the expression evaluator invokes an overridden implementation of a ToString method to facilitate presentation of debug information. However, it is well known in the art that ToString methods return a string representation of an object and are used primarily for debugging. It would have been obvious to one having ordinary skill in the art at the time the invention was made to include an overridden implementation of a ToString method for debugging for the convenience of displaying the output for a program and simplifying the process of debugging the program.

As to claim 6, Bates in view of Bates_2 teaches the limitations of claim 1, but does not specifically teach that the expression evaluator employs a result of the overridden ToString method as a value to be displayed for an object, field, property, or combinations thereof. However, it is well known in the art that ToString methods return a string representation of an object and are used primarily for debugging. It would have been obvious to one having ordinary skill in the art at the time the invention was made to include an overridden implementation of a ToString method for debugging and use the returned result as a display value for the convenience of displaying the output for a program and simplifying the process of debugging the program.

As to claim 7, Bates also teaches the system of claim 1, wherein at least the one or more attributes determines at least one of how and whether a type or member is displayed (see page 6, paragraph [0064]: *at step 614, the debugger determines whether any attributes are set for the variable and paragraph [0065]: that is, the debugger determines whether a field is to be displayed*).

As to claim 13, Bates also teaches the system of claim 1, wherein at least one of the one or more attributes specifies what is displayed for a class and/or field (see page 6, paragraph [0064]: *at step 614, the debugger determines whether any attributes are set for the variable and paragraph [0065]: the debugger determines whether a field is to be displayed*).

As to claim 14, Bates in view of Bates_2 teaches the system of claim 13, wherein an argument to the attribute comprising a string that is displayed in a value column for an instance of the class and/or field (see page 6, paragraph [0064]: *if any attributes are set for the variable, then processing proceeds to step 616 where the appropriate attribute indicator is associated with the variable value*).

As to claim 15, Bates also teaches wherein the argument is associated with one of a field, a property or a method, or combinations thereof (see Bates: page 6, paragraph [0065]: *the debugger determines whether the variable value is associated with a field of a record*).

As to claim 16, Bates also teaches the system of claim 11 further comprising an attribute cache directory (see FIG. 1: 150 and associated text) that stores an attribute associated with the computer software application, the expression evaluator employing the stored attribute to present debug information (see page 3, paragraph [0033]: *the database management system includes a database which may be a variety of repositories… the database provides one example of an external data source for external comments and other variable information*).

As to claim 17, Bates teaches a computer-implemented method for facilitating debugging a computer software application using attributed debugging, the method comprising:

determining whether a process in a computer software application has an attribute attached thereto (see page 6, paragraph [0064]: *at step 614, the debugger determines whether any attributes are set for the variable*),

if the process has an attribute attached thereto, evaluation the attribute according to a definition of the attribute to display debug information (see paragraph [0065]: *the debugger determines whether a field is to be displayed*), wherein the evaluation of the definition of the attribute (*i.e. evaluation of Call attribute to be on or off*, see Fig.3 and associated text) determines that a value (*i.e. character C*, see Fig.7, 710 and associated text) to be shown for a field or property (*i.e. flyover text box,* see FIG.7, 710 and associated text, e.g. [0067] lines 12-14) of the software application is the result of an evaluation of an expression contained in the code of the software application (*i.e. determination that the Call attribute is to be on*, see Fig. 3 and associated text), which value is different from the value that would be shown for the field or property when the corresponding attribute is removed (*i.e. attribute is turned off*, see Fig. 3 and associated text and *therefore not displayed in the flyover text box*, see Fig.7,710 and associated text)

Bates does not specifically teach displaying the debug information in a developer-customizable format according to declarative indication included in the attribute. However, in an analogous art, Bates_2 is cited to teach where debug information is presented in a developer-customizable format (see column 2, lines 8-12 and lines 28-37 and FIG. 4 and associated text, e.g. column 3, lines 49-55). It would have been obvious to one having ordinary skill in the art at the time of the invention to combines the teachings of Bates and Bates_2 for the purpose of providing an improved debugger having a mechanism when debugging programs that provides custom record displays, where only user selected fields of records or variables are displayed, as disclosed by Bates_2 (see column 1, lines 53-56).

As to claim 18, Bates in view of Bates_2 does not specifically teach determining whether a ToString method has been overridden and,
invoking the overridden ToString method to facilitate debugging, if the ToString method has been overridden. However, it is well known in the art that ToString methods return a string representation of an object and are used primarily for debugging. It would have been obvious to one having ordinary skill in the art at the time the invention was made to include an overridden implementation of a ToString method for debugging and use the returned result as a display value for the convenience of displaying the output for a program and simplifying the process of debugging the program.

As to claim 19, Bates also teaches a computer readable storage medium having stored thereon computer executable instructions for carrying out the method of claim 17 (see FIG. 1 and associated text).

As to claim 21, Bates teaches a computer readable storage medium (see [0023]) storing computer executable components of an attributed debugging system, the attributed debugging system comprising:

a debugger component (see Fig.1, 123 and associated text) that allows the manipulation of view data in the debugger (see Fig.6B and associated text, e.g. [0067] lines 1-5), and

an expression evaluator component (see Fig.1, 126 and associated text) that employs an attribute definition to evaluate an attribute associated with the computer software application to present debug information associated with the computer software application to a developer (see [0064]: *at step 614, the debugger determines whether any attributes are set for the variable; if any attributes you set for the variable, then processing proceeds to step 616 where the appropriate attribute indicator (e.g. G,S,I,R,C,P) for each set attribute is associated with the variable value and paragraph [0065]: that is, the debugger determines whether a field is to be displayed*) wherein the employment of the attribute definition (*i.e. evaluation of Call attribute to be on or off*, see Fig.3 and associated text) determines that a value (*i.e. character C*, see Fig.7, 710 and associated text) to be shown for a field or property (*i.e. flyover text box,* see FIG.7, 710 and associated text, e.g. [0067] lines 12-14) of the software application is the result of an evaluation of an expression contained in the code of the software application (*i.e. determination that the Call attribute is to be on*, see Fig. 3 and associated text), which value is different from the value that would be shown for the field or property when the corresponding attribute is removed (*i.e. attribute is turned off*, see Fig. 3 and associated text and *therefore not displayed in the flyover text box*, see Fig.7,710 and associated text).

Bates does not specifically teach wherein the debug information is displayed in a developer- customizable format in accordance with the attribute definition and in a developer- customizable data window. However, in an analogous art, Bates_2 is cited to teach where debug information is presented in a developer-customizable format (see column 2, lines 8-12 and lines 28-37 and FIG. 4 and associated text, e.g. column 3, lines 49-55). It would have been obvious to one having ordinary skill in the art at the time of the invention to combines the teachings of Bates and Bates_2 for the purpose of providing an improved debugger having a mechanism when debugging programs that provides custom record displays, where only user selected fields of records or variables are displayed, as disclosed by Bates_2 (see column 1, lines 53-56).

7. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Bates et al (US Patent Application Publication 2003/0221185 A1) in view of Bates et al (US Patent 7,251,808 B2, referred to as Bates_2) and Kobayashi et al (US Patent 6,633,888 B1).

As to claim 22, Bates teaches a computer-implemented attributed debugging system for debugging a computer software application, comprising:

a debugger that facilitates debugging of a computer software application (see FIG.1: 123 and associated text, e.g. [0040]),

a debuggee that includes one or more attributes associated with the computer software application, which attributes are employed by the debugger to facilitate debugging of the software application (see Fig,1 119 and associated text, e.g. [0037] lines 8-10) wherein at least one of the attributes is a DebuggerBrowseableAttribute that is employed to determine whether and how a type or member is displayed in a data window (See Fig.6B and associated text, e.g. [0067] lines 1-5 and , and wherein at least of the attributes is a DebuggerDisplayAttribute that is employed to control what is displayed for a class or field in the data window (see [0065] lines 1-4 and Fig.6B and associated text, e.g. [0067] lines 1-5), and

an expression evaluator (see Fig.1, 126 and associated text) that evaluates the one or more attributes associated with the computer software application according to an attribute definition (see Fig. 3 and associated text, e.g. [0047] lines 23-31), and presents debug information associated with the computer software application in accordance with the attribute definition (see Fig.6B, 628 and associated text, e.g. [0067] lines 8-10), wherein the evaluation of at least one of the one or more attributes (*i.e. evaluation of Call attribute to be on or off*, see Fig.3 and associated text) determines that a value (*i.e. character C*, see Fig.7, 710 and associated text) to be shown for a field or property (*i.e. flyover text box,* see FIG.7, 710 and associated text, e.g. [0067] lines 12-14) of the software application is the result of an evaluation of an expression contained in the code of the software application (*i.e. determination that the Call attribute is to be on*, see Fig. 3 and associated text), which value is different from the value that would be shown for the field or property when the corresponding attribute is removed (*i.e. attribute is turned off*, see Fig. 3 and associated text and *therefore not displayed in the flyover text box*, see Fig.7,710 and associated text).

Bates does not specifically teach wherein the attribute definition declaratively indicates the developer-customizable format in which the debug information is displayed in a developer-customizable data window. In an analogous art, however, Bates_2 is cited to teach where debug information is presented in a developer-customizable format (see column 2, lines 8-12 and lines 28-37 and FIG. 4 and associated text, e.g. column 3, lines 49-55). It would have been obvious to one having ordinary skill in the art at the time of the invention to combines the teachings of Bates and Bates_2 for the purpose of providing an improved debugger having a mechanism when debugging programs that provides custom record displays, where only user selected fields of records or variables are displayed, as disclosed by Bates_2 (see column 1, lines 53-56).

Neither Bates nor Bates_2 specifically teach wherein at least one of the attributes is a DebuggerTypeProxyAttribute that is employed to specify the display proxy of a type. In an analogous art, however, Kobayashi is cited to teach a DebuggerTypeProxyAttribute that is employed to specify the display proxy of a type (see col.5 lines 1-12) It would have been obvious to one having ordinary skill in the art at the time of the invention to have combined the teaches of Bates and Bates_2 with those of Kobayashi in order to provide an improved method of testing software components within a visual builder that would increase programmer efficiency and reduce the time needed to develop code, as disclosed by Kobayashi (see col.4 lines 43-50).

8. Claims 2-4 and 8-12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bates et al (US Patent Application Publication 2003/0221185 A1) in view of Bates et al (US Patent 7,251,808 B2, referred to as Bates_2) as applied to claims 1 and 7 above, and further in view of Dandoy (US Patent Application Publication 2004/0230954 A1).

As to claim 2, Bates in view of Bates_2 teaches the limitations of claim 1, but neither specifically teaches that the expression evaluator evaluates an expression associated with a particular programming language. However, in an analogous art, Dandoy teaches a system for debugging a software application that displays the contents and properties of objects and determines what events are emitted by objects (see paragraph [0046]). It would have been obvious to one having ordinary skill in the art to combine the teachings of Bates and Bates_2 with that of Dandoy because the systems and methods of Dandoy's invention can be configured to debug software that was created with other programming languages besides Java or different variants of Java (see page 6, paragraph [0048]).

As to claim 3, Bates in view of Bates_2 teaches a debugger and an expression evaluator, but neither specifically teaches the programming language is at least one of C#, J# or Visual Basic.Net. However, in an analogous art, Dandoy teaches a system for debugging a software application that displays the contents and properties of objects and determines what events are emitted by objects (see paragraph [0046]). It would have been obvious to one having ordinary skill in the art to combine the teachings of Bates and Bates_2 with that of Dandoy because the systems and methods of Dandoy's invention can be configured to debug software that was created with other programming languages, including Java, HTML, C#, C++, and C (see page 6, paragraphs [0048] and [0049]).

As to claim 4, Bates in view of Bates_2 teaches a debugger and an expression evaluator, but neither specifically teaches a plurality of expression evaluators, each expression evaluator associated with a particular programming language. However, in an analogous art, Dandoy teaches a system for debugging a software application that displays the contents and properties of objects and determines what events are emitted by objects (see paragraph [0046]). It would have been obvious to one having ordinary skill in the art to combine the teachings of Bates and Bates_2 with that of Dandoy because the systems and methods of Dandoy's invention can be configured to debug software that was created with other programming languages, including Java, HTML, C#, C++, and C (see page 6, paragraphs [0048] and [0049]).

As to claim 8, Bates in view of Bates_2 teaches the limitations of claim 7, but neither specifically teaches the attribute employing an enumeration. However in an analogous art, Dandoy teaches how the debug agent is configured to collect execution data relating to the graphical user interface, which includes object properties, events, and runtime states (see page 2, paragraph [0018]). It would have been obvious to one having ordinary skill in the art at the time of the invention to combine the teachings of Bates and Bates_2 with that of Dandoy for the advantage of gaining a more efficient debugging system that does not require the source code of the application being debugged to be modified and saving programmers and developers valuable time.

As to claim 9, Dandoy further teaches one enumeration value associated with an indication that the type or member should not be displayed to the developer (see page 3, paragraph [0025], *the debug agent can determine the current state of the selected window and change its properties such that the window is hidden*).

As to claim 10, Dandoy further teaches one enumeration value associated with an indication that if a type is hierarchical, it should be expanded by default (see page 5, paragraph [0046]: *at any point during debugging, a hierarchy of objects within the interface can be determined and displayed; the hierarchy can be displayed automatically*).

As to claim 11, Dandoy further teaches one enumeration value associated with an indication that a type should not be expanded by default (see page 3, paragraph [0024], *debugging requests may include a request to monitor events associated with an object... or request to hide or show an object*).

As to claim 12, Dandoy further teaches one enumeration value associated with an indication that a target element itself should not be shown, but should instead be automatically expanded to have its member(s) displayed (see page 5, paragraph [0046], *at any point during debugging, a hierarchy of objects within the interface can be determined and displayed; the hierarchy can be displayed automatically*).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to CHENECA P. SMITH whose telephone number is (571)270-1651. The examiner can normally be reached on Monday-Friday 7:00-4:30 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Cheneca P Smith/
Examiner, Art Unit 2192
9/29/09

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192